Desenvolvimento de Jogos em HTML: Uma abordagem sobre o *Canvas*

Lucélio Dias de Azevedo[1], Maurício J.V.Amorim[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia Fluminense
Av. Dr. Siqueira, 273 – Campos dos Goytacazes – RJ – Brazil

luceliodias@msn.com, amorim@cefetcampos.br

Abstract *This article demonstrates the basic knowlege for game development on HTML5 using the Canvas element. The focuses is on the demonstration of a practical example that helps the reader to understand the basic techniques of game development on HTML5. The article demonstrates an overview about games, HTML and a code example using the Canvas element.*

Resumo *Este artigo demonstra o conhecimento básico para desenvolvimento de jogos em HTML5 utilizando o elemento Canvas e concentra-se na demonstração de um exemplo prático que ajuda o leitor a entender as técnicas de desenvolvimento.*

Palavras Chaves*: Jogos, HTML5, Canvas.*

1. Introdução

Um jogo eletrônico permite ao desenvolvedor criar seu próprio mundo e dentro dele regras que podem desafiar o usuário, podendo trazer grande interação com o mesmo. O ambiente criado pode tanto retratar uma realidade, como pode também utilizar-se de conceitos fantasiosos. Desta maneira, fica claro que o limite no desenvolvimento de jogos é a criatividade da pessoa e/ou equipe que produzirá o jogo.

A grande vantagem do *HyperText Markup Language* (HTML) é a portabilidade entre as diferentes plataformas e um jogo desenvolvido de acordo com os padrões definidos pelo *World Wide Web Consortiun* (W3C) pode ser executado diretamente em qualquer navegador atual, seja ele o Mozilla Firefox, Google Chrome, Internet Explorer, Opera, Safari sem alteração do código fonte.

Este artigo utilizará o elemento Canvas, disponível na especificação do HTML5 como base para desenvolvimento de um protótipo de jogo, buscando demonstrar uma alternativa no desenvolvimento de jogos multi-plataforma.

2. Jogos

Jogos eletrônicos estão entre as maiores fontes de entretenimento do mundo. Do lançamento do primeiro jogo em longínquos 1958 até os dias de hoje, muita coisa mudou no mercado de jogos eletrônicos. As diferentes plataformas lançadas, bem como o aumento de desempenho dos consoles e computadores tornaram a área de

desenvolvimento jogos um nicho extremamente atraente e que movimenta bilhões de dólares anualmente.

A pesquisa realizada em (VIEIRA, AZEREDO, NETO; 2006) mostra que a utilização de jogos eletrônicos iniciou-se no ano de 1958, mais precisamente em Nova Iorque, sendo criada uma pequena "simulação" de uma partida de tênis, recebendo o nome de "Tennis Programming" ou popularmente chamado de "Tennis for Two", e é considerado o precursor dos jogos eletrônicos.

O início da década de 70 marcou o lançado do console Odissey 100 pela empresa Magnavox sendo a primeira investida dos jogos eletrônicos no mercado popular. O console fez pouco sucesso porque mesmo para sua época, era simplório demais.

Imagem 2: Console Odissey

Em 1972, Nolan Bushnell, um dos jovens acadêmicos pesquisadores do MIT, deixa seu emprego e funda sua própria empresa, a Atari, que teve como primeiro produto comercializado um jogo bastante simples chamado "Pong" que era constituído por dois traços nas laterais da tela e um ponto, considerado a bola. Nos anos seguintes, a partir do sucesso da versão doméstica de "Pong", produzido em parceria com a as lojas Sears, inúmeras empresas resolveram lançar produtos similares ao da Atari, contudo, poucos tiveram algum sucesso no mercado, tendo a Atari se tornado referência para o mercado doméstico.

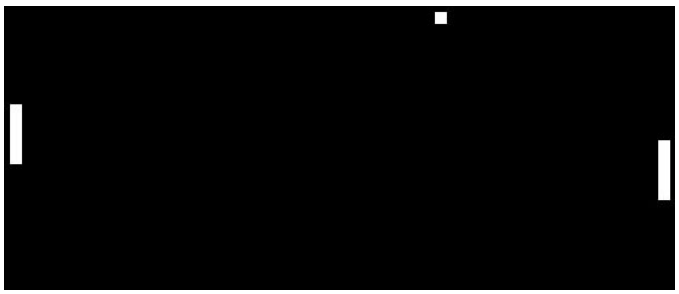

Imagem 3: Jogo Pong.

A velocidade da evolução da computação foi benéfica ao mundo dos jogos, pois a utilização das novas tecnologias possibilita na maioria das vezes aumentarmos o nível de interação com o usuário e por consequência, melhora na qualidade do jogo.

A classificação dos tipos de jogos depende de sua interface e interação com o usuário, e leva em conta seu objetivo principal e são definidas como:

- Aventura: jogos que se baseiam em histórias, geralmente voltados a resolver problemas ou solucionar mistérios, onde o jogador tem que explorar um vasto e complexo mundo.
- Ação: jogos nos quais os jogadores têm que se valer de seus reflexos e destrezas para responder rapidamente aos eventos ocorridos em cena.
- Role Playing Game (RPG): este estilo de jogo se baseia no RPG de mesa, na qual tem como idéia central a representação de um personagem através de um conjunto de regras.
- Estratégia: este estilo de jogo tem como foco principal o gerenciamento de recursos para atingir um determinado objetivo. Para que tal objetivo seja alcançado o jogador deve elaborar um plano, se preocupando em como obter os recursos, e como utilizá-los.
- Simulação: jogos que simulam condições, ambientes e situações do mundo real. Além de entreter, eles podem ser usados também para auxiliar os treinamentos em equipamentos complexos, como carros, aviões, submarinos, entre outros.
- Esportes: são jogos que simulam os esportes, tanto os coletivos quanto os individuais. Tentam reproduzir os movimentos dos atletas para o mundo virtual dos games da forma mais realística possível.
- Luta: Estes são jogos onde um jogador escolhe um personagem para enfrentar outro oponente (jogador humano ou PNJ). Cada jogador executa uma sequência de comandos para acionar ações como socar, chutar, saltar, utilizar um ataque especial, entre outras.
- Plataforma: é um gênero de jogos para videogame onde o personagem tem que escalar ou pular plataformas enquanto enfrentam adversários e reúnem itens para completar o jogo.
- Educacionais: jogos com o intuito de educar e adicionar informações aos jogadores ao mesmo tempo em que eles se divertem. É voltado para o público infantil, utilizando uma linguagem bem simples e personagens bastante carismáticos.
- Puzzle: jogos voltados para desafiar o intelecto do jogador, através de soluções de problemas e enigmas, sem contexto de história.
- Massive Multiplayer Online (MMO): s jogadores devem estar conectados em uma rede (ou internet) e podem utilizar qualquer um dos conceitos citados anteriormente.

Um jogo pode utilizar um ou mais estilos diferentes para sua classificação. Neste contexto, são levadas em consideração as mesmas regras de classificação, como interface e interação como usuário.

3. HTML5

A evolução das tecnologias nos oferece diversos caminhos para o desenvolvimento de jogos e atualmente *HyperText Markup Language* (que possui o acrônimo HTML) é um deles. O HTML é uma linguagem de marcação para descrição de documentos de internet (*web pages*) que utiliza *tags* para descrever os diferentes conteúdos dentro de um documento (W3SCHOOLS, 2015a).

A *world wide web* começou a ganhar vida no Laboratório Europeu de Física de Partículas (CERN), onde, em 1989, Tim Berners-Lee, que trabalhava em uma seção de serviços de computação do laboratório apresentou o conceito. Foi sugerido por Tim que documentos poderiam fazer referências cruzadas de uma pesquisa para outra, criando uma teia (web) de informações. O uso de um elemento âncora com o atributo HREF foi o que fez a invenção de Tim tão útil. Nascia então de forma muito simples, mas eficiente para seu propósito na época, o HTML (W3C, 1998).

De 1991 em diante o HTML obteve muitas melhorias, pois o padrão foi mantido aberto e muitos entusiastas sugeriram melhorias ao longo do tempo. Os primeiros rascunhos do HTML5 foram criados em 2008 e o mesmo é descrito por (W3SCHOOLS, 2015a) como a nova versão do HTML4 que define novas APIs que formarão base da arquitetura web. A nova especificação disponibiliza novas *tags* e modifica a função de outras (W3SCHOOLS, 2015b), trazendo uma nova forma de escrevermos e organizar o código e as informações da página, utilizando uma forma mais semântica e com menos código. Isso por consequência traz mais interatividade sem a necessidade de instalação de plugins e perda de desempenho.

Até o lançamento do HTML5 e CCS3, todas as idéias listadas em especificações deveriam ser testadas e desenvolvidas para só depois serem publicadas para uso dos browsers e desenvolvedores, método este alterado com as duas tecnologias sendo divididas em módulos e a comunidade e fabricantes não precisam esperar um padrão ser publicado (W3C, 2010).

4. Canvas

O Canvas é o novo elemento (*tag*) responsável pela renderização gráfica nos navegadores. As animações para ele geralmente são escritas utilizando Javascript e pode ser usado para renderizar texto, imagens, gráficos, linhas, gradientes e outros efeitos de forma dinâmica. Navegadores mais recentes dão suporte a utilização da GPU para renderização do Canvas, usando o Direct2D, acelerando o processamento e deixando a CPU para cuidar de outras tarefas com mais eficiência (MICROSOFT, 2013). O novo elemento Canvas permite renderização de gráficos, gráficos do jogo, ou outras imagens visuais em tempo real (*on the fly*) (SHANKAR, 2012).

O uso do elemento Canvas será demonstrado com a criação da base de um jogo no estilo “Pong” (MAILSON, 2013). Um arquivo HTML deve possuir a tag

<canvas>, e para manter a organização, os scripts do jogo estarão no arquivo pong.js. No arquivo *Javascript* estará concentrada toda a base lógica do jogo. A estrutura do arquivo HTML pode ser codificada como:

```
<!DOCTYPE html>
<html>
<head>
    <title>Pong</title>
    <script defer src="pong.js"></script>
    <style type="text/css">
        #game {
            background-color: #353535;
        }
    </style>
</head>
<body>
    <canvas id="game" width="512" height="256"></canvas>
</body>
</html>
```

Imagem 3.3: Base para o HTML de exemplo proposto no trabalho.

A tag *defer* é utilizada na chamada do script é feita para garantir que o arquivo será executado somente depois de carregado todo o arquivo HTML. O jogo possuirá duas funções principais, **atualizar()** e **desenhar()** e o trecho de código deve ser escrito no arquivo “pong.js”:

```
function Game() {
    var canvas = document.getElementById("game");
    this.width = canvas.width;
    this.height = canvas.height;
    this.context = canvas.getContext("2d");
    this.context.fillStyle = "white";
}

Game.prototype.desenhar = function()
{
    this.context.clearRect(0, 0, this.width, this.height);
    this.context.fillRect(this.width/2, 0, 2, this.height);
};

Game.prototype.atualizar = function()
{
    if (this.paused)
        return;
};
```

Figura 3.4: Criação do jogo.

Na função Game(), existem duas variáveis importantíssimas para a criação do jogo, que são:

- canvas: guarda um ponteiro para o objeto canvas que tem a *id* “*game*”.

- context: guarda o contexto 2D para o elemento canvas utilizado.

O objeto de contexto provê um grande número de métodos que podem ser utilizados para desenhar algo na tela. Isto inclui métodos para (SHAKTAR, 2012):

- Desenhar retângulos;
- Desenhar caminhos complexos (linhas, arcos e etc.);
- Desenhar texto;
- Customizar estilos de desenho (cores, *alpha*, texturas e outros);
- Desenhar imagens;
- Transformar e rotacionar;

O *Javascript* disponibiliza a propriedade *prototype* para adicionar métodos e outras propriedades à um objeto (W3SCHOOLS, 2015b). Outras funções utilizadas no trecho de código proposto são:

- clearRect: limpa a área do retângulo especificado e o torna totalmente transparente. No nosso caso, ele demarca e limpa a área de um retângulo que vai de (0,0) até o tamanho do elemento canvas;
- fillRect: Desenha um retângulo na posição com o tamanho especificado;

A base para o utilizará um *loop* principal que faz chamada as funções desenhar() e atualizar() do objeto game. O trecho de código pode ser parecido com:

```
var game = new Game();

function LoopPrincipal() {
    game.atualizar();
    game.desenhar();
    requestAnimationFrame(LoopPrincipal);
}

LoopPrincipal();
```

Figura 3.5: Novo jogo e *loop* principal.

O *requestAnimationFrame* fornece uma maneira mais suave e eficiente para criar páginas *web* com animações, chamando o *frame* de animação apenas quando o sistema está pronto para desenhá-lo. Como resultado, tudo estará perfeitamente alinhado com o intervalo de desenho que navegador precisa e isso utiliza somente a quantidade adequada de recursos de memória e processamento (W3C, 2013).

Para criar os jogadores, será escrita uma classe de nome *Paddle* e definida uma função **desenhar()** para a mesma:

```
function Paddle(x,y) {
    this.x = x;
    this.y = y;
    this.width = 2;
    this.height = 28;
    this.score = 0;
}

Paddle.prototype.desenhar = function(p)
{
    p.fillRect(this.x, this.y, this.width, this.height);
};
```

Figura 3.6: Classe e função para criação de jogadores.

A instanciação e inserção dos jogadores na tela do jogo devem ser realizadas na classe Game:

```
this.p1 = new Paddle(5, 0);
this.p1.y = this.height / 2 - this.p1.height / 2;
this.p2 = new Paddle(this.width - 5 - 2, 0);
this.p2.y = this.height / 2 - this.p2.height / 2;
```

Figura 3.7: Código para criação dos jogadores.

E na função **desenhar()** da classe Game é realizada a chamada a função **desenhar()** da classe Paddle:

```
this.p1.desenhar(this.context);
this.p2.desenhar(this.context);
```

Figura 3.8: Código que desenha os jogadores na tela.

Ao executar o código proposto, será visualizado algo parecido com:

Figura 3.9: Visualização da base do jogo proposto.

Para finalizar a proposta do artigo, será incluída a movimentação dos jogadores utilizando as teclas S e W pra movimentar o jogador 1 e setas para cima e para baixo para movimentar o jogador 2. Deverá ser criado um ouvidor (*listener*), responsável por capturar os eventos do teclado:

```
function KeyListener() {
    this.pressedKeys = [];

    this.keydown = function (e) {
        this.pressedKeys[e.keyCode] = true;
    };

    this.keyup = function (e) {
        this.pressedKeys[e.keyCode] = false;
    };

    document.addEventListener("keydown", this.keydown.bind(this));
    document.addEventListener("keyup", this.keyup.bind(this));
}

KeyListener.prototype.isPressed = function (key) {
    return this.pressedKeys[key] ? true : false;
};

KeyListener.prototype.addKeyPressListener = function (keyCode, callback) {
    document.addEventListener("keypress", function (e) {
        if (e.keyCode == keyCode)
            callback(e);
    });
};
```

Figura 3.10: Adicionando os *keylisteners* ao jogo.

Para dar o movimento aos jogadores, é necessário criar uma nova instância do objeto KeyListener na classe Game e o código será incluído antes da definição de criação dos jogadores:

```
this.keys = new KeyListener();
```

Figura 3.11: Instanciando o keylistener.

No método **update()** do game, deverá ser incluído o código que verifica qual tecla está sendo pressionada:

```
if (this.keys.isPressed(83)) {
    this.p1.y = Math.min(this.height - this.p1.height, this.p1.y + 4);
} else if (this.keys.isPressed(87)) {
    this.p1.y = Math.max(0, this.p1.y - 4);
}

if (this.keys.isPressed(40)) {
    this.p2.y = Math.min(this.height - this.p2.height, this.p2.y + 4);
} else if (this.keys.isPressed(38)) {
    this.p2.y = Math.max(0, this.p2.y - 4);
}
```

Figura 3.12: Verificação das teclas pressionadas.

O exemplo acima foi dado visando explicar funcionamento básico de um jogo utilizando o Canvas. Dependendo da complexidade da jogo que se deseja criar, é certo que necessitarão por parte do leitor, mais estudos sobre o tema.

5. Resultados Computacionais

O resultado obtido com este levantamento tem como grande contribuição a demonstração das técnicas de base no desenvolvimento de jogos em HTML5. Fica claro por meio do exemplo citados que a especificação da API pode ser utilizada de muitas maneiras e a principal limitação é a criatividade da equipe desenvolvedora.

5.1 Multi-plataforma

Para criar jogos em HTML5, é comum a utilização da linguagem Javascript. Como é uma linguagem interpretada, permite que os testes sejam realizados na medida em que o código é escrito. O desenvolvedor necessita apenas abrir um navegador e rodar o código.

A capacidade de funcionar em qualquer navegador é uma das situações que mais chama a atenção nesta abordagem, pois os principais navegadores estão disponíveis para os sistemas operacionais mais utilizados, sejam ele desktop ou mobile. Logo, um jogo desenvolvido em HTML5 pode atender a praticamente todos os usuários de internet, e para isso basta que os mesmos mantenham seu navegador atualizado com a versão mais recente disponível.

Imagem 5: Principais navegadores do mercado.

6. Considerações Finais

Este estudo apresenta uma alternativa para desenvolvimento de jogos bidimensionais utilizando HTML5. O artigo utiliza de maneira didática a exemplificação de um código simples, mas determinantes para base de aprendizado das técnicas fundamentais de desenvolvimento. Tem-se, assim, como grande contribuição do artigo, a demonstração das técnicas de desenvolvimento de jogos em HTML5, servindo de importante fonte de conhecimento para aprendizado daqueles que pensam em iniciar os estudos no desenvolvimento de jogos.

Um passo importante que pode se utilizado em trabalhos futuros seria a criação da bola e aplicação da física para que ela possa ser rebatida pelos jogadores. Em um segundo momento, fazer uso de algum um *framework* que fornece controle sobre as tarefas comuns da área de jogos. Tipicamente, *frameworks* de jogos trazem suporte à controle de gráficos, física, detecção de colisão, suporte à áudio, animações, gerenciamento de memória e muitas outras tarefas. Como exemplo, pode ser utilizado o PHASER (PHASER, 2015).

Imagem 7: Logotipo do *framework* Phaser.

Referências

MAILSON. Simple pong game using HTML5 and canvas .2013. Disponível em: <http://blog.mailson.org/2013/02/simple-pong-game-using-html5-and-canvas/>. Acesso em: 13 jul. 2015.

MICROSOFT. HTML5 - Guia do desenvolvedor para o HTML5 Canvas. 2013. Disponível em: <https://msdn.microsoft.com/pt-br/library/dn151487.aspx>. Acesso em: 10 jul. 2015.

PHASER. Phaser - A fast, fun and free open source HTML5 game framework. 2015. Disponível em: <http://phaser.io>. Acesso em: 09 jul. 2015.

SHANKAR, Aditya Ravi. Pro HTML5 Games. 1 ed. Apress, 2012. 364p. ISBN: 978-1-4302-4710-4

STATCOUNTER. StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share. 2015. Disponível em: <http://gs.statcounter.com/>. Acesso em: 13 jul. 2015.

VIEIRA, Pedro Sampaio; AZEREDO, Thiago Ribeiro de; SANTOS, Antônio Alves dos. *Inteligência Artificial aplicada a tomada de decisão em Jogos Eletrônicos*. 76f. Monografia (Bacharelado em Sistemas de Informação) - Universidade Candido Mendes, Campos, Rio de Janeiro, 2006.

W3SCHOOLS. Introdution to HTML. 2015a. Disponível em: http://www.w3schools.com/html/html_intro.asp. Acesso em: 10 jul. 2015.

_____. JavaScript prototype Property. 2015b. Disponível em: <http://www.w3schools.com/jsref/jsref_prototype_math.asp>. Acesso em 21 jul. 2015.

W3C. A history of HTML. 1998. Disponível em: <http://www.w3.org/People/Raggett/book4/ch02.html>. Acesso em: 10/07/2015.

_____. HTML5 Curso W3C Escritório Brasil. 2010. Disponível em: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>. Acesso em: 10 jul. 2015.

_____. Timing control for script-based animations. 2013. Disponível em: <http://www.w3.org/TR/animation-timing/>. Acesso em: 21 jul. 2015.